COBB

INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS Versions 5 & 6

July 1994
Vol. 5 No. 7

Opening a highlighted file in the Editor from the DOS Shell

VERSION 5.0 & 6.x

Many use the DOS Shell primarily for file maintenance and others use it for its multitasking capabilities. If you use the Shell for running programs, you want to make that task as easy as possible. In this article, we'll show how to add a program item to the Shell that will let you open the currently highlighted file in the DOS Editor. (In "Opening and Closing Files in the MS-DOS Editor" on page 7, we show you several other ways to open files in the Editor.)

Running programs in the DOS Shell

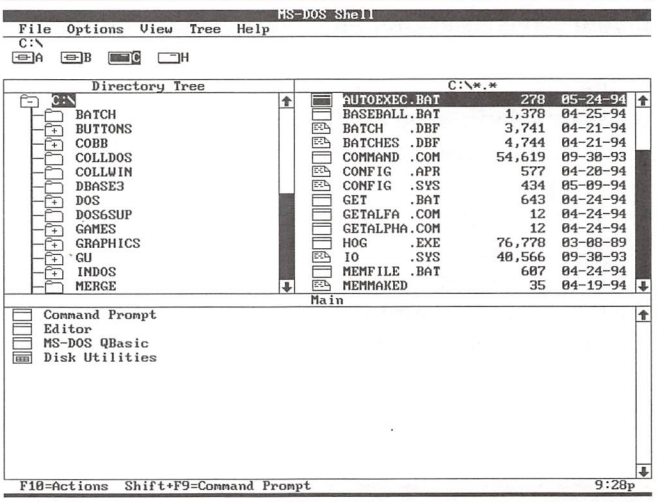
The DOS Shell offers several methods for running programs such as the DOS Editor. For one, you can open the Shell's File menu, select Run..., and enter the name of the program in the Command Line text box. For example, you can use the keyboard to start the Editor by pressing [Alt]F and then R to open the Run dialog box, typing *edit*, and then pressing [Enter].

You can also run a program in the DOS Shell by selecting its executable file. For example, to start the Editor using a mouse, you click the DOS directory in the directory tree and then double-click the filename EDIT.COM in the file list window.

Another way to start the Editor is to press [Shift][F9] to shell out to the command prompt. From there, you run the commands and programs you want, and then type *exit* and press [Enter] to return to the Shell. To open the Editor, you enter *edit* at the prompt. After you finish using the Editor, you type *exit* and press [Enter] to return to the Shell.

All these methods are instantly available but require lots of keypresses. The DOS Shell provides the capability to set up shortcuts for programs and commands you run frequently. By attaching commands to icons in the program list, you can run a program just by selecting its icon. The DOS Shell provides several default icons, called *program items*, for frequently used DOS commands and utilities—including a default Editor item we'll improve upon later. The program list appears at the bottom of the DOS Shell screen, as shown in Figure A. (If your screen doesn't display the directory tree, file list, and program list as shown in Figure A, select the Program/file lists option from the View menu.

Figure A



The default DOS Shell screen displays windows containing the directory tree, file list, and Main program list.

A closer look at the program list

The program list contains program items and program groups. A *program item* contains startup instructions for a program. You can think of a program item as a short batch file that you can execute simply by double-clicking the item or by highlighting it and

IN THIS ISSUE

pressing [Enter]. For example, you can start the DOS Editor by pressing [Tab] to place the cursor in the Main window, pressing the ↓ key until *Editor* is highlighted, and pressing [Enter]. When you follow these steps, the Shell executes the command

```
EDIT %1
```

The %1 parameter opens a dialog box that prompts you for the name of a file to edit. You can add a filename to the text box or leave it blank. Once you press [Enter], the Editor opens the file you specified or an untitled screen.

A program item is represented by the (▭) icon if your screen is in Graphics mode. If your screen is in Text mode, a program item appears without an icon in the program list. (You can switch to Graphics mode by selecting the Display... option on the Options menu and selecting one of the Graphics choices. We use Graphics 34 lines, Medium Resolution.)

A *program group* is a collection of program items. Program groups can also contain other program groups. For example, the default Main program group consists of the Command Prompt item, the Editor item, the MS-DOS QBasic item, and the Disk Utilities group. Program groups are represented by the (▤) icon in Graphics mode. In Text mode, the program group names appear in square brackets—[Disk Utilities], for example.

You can add groups and program items to the program list—we'll do that next. The default Editor program item uses a dialog box in which you can specify which file you want to open. We're going to go one better—we'll add a program item that will start the Editor and open the file whose name is highlighted in the file list.

Adding an Edit Highlighted File program item

We'll add the new program item to the Main program group. In order to add a program item to any group, you must first open that group. So, make sure that your cursor is in the program list and that the shaded bar at the top of the program list window contains the word *Main*. If you see a different heading, find the Main group entry in the program list and double-click it or highlight it and press [Enter].

Next, open the File menu and choose the New... option to open the New Program Object dialog box, shown in Figure B. We'll add a program item, so use your mouse or your ↑ or ↓ key to select the Program Item option and then press [Enter]. When you do, the Add Program dialog box opens.

INSIDE DOS

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices: Domestic: $49/yr ($6.00 each)
Outside US: $69/yr ($8.50 each)

Phone: Toll free: (800) 223-8720
Local: (502) 493-3300
Customer Relations Fax: (502) 491-8050
Editorial Department Fax: (502) 491-3433

Address: You may address tips, special requests, and other correspondence to

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

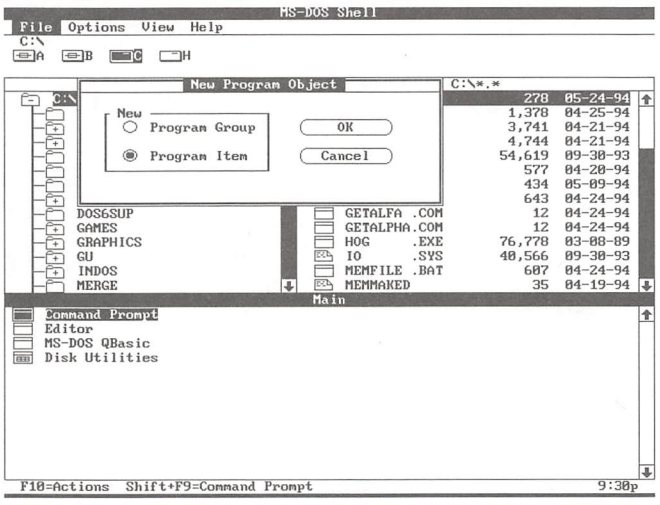
For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster: Second class postage is pending in Louisville, KY. Send address changes to
Inside DOS
P.O. Box 35160
Louisville, KY 40232

Authorized Canada Post International Publications Mail (Canadian Distribution) Sales Agreement #XXXXXX CANADA GST #123669673. Send returns to Canadian Direct Mailing Sys. Ltd., 920 Mercer Street, Windsor, Ontario, N9A 7C2. Printed in the USA.

Figure B



You choose New... from the File menu to open the New Program Object dialog box.

You enter several pieces of information in the Add Program dialog box. You must enter the program item's name, which can contain up to 23 characters, as well as the necessary startup commands. Optionally, you can enter a startup directory, which is useful in applications in which you keep documents in a different directory from the one that holds the file that starts the program. Also, you can add a shortcut key you can use to run the program from anywhere inside the DOS Shell, even from another task. If security is an issue, you can add a password of up to 20 characters in order to limit access to a program item or group.

To create our Edit Highlighted File program item, enter the text

```
Edit Highlighted File
```

in the Program Title text box. This text will display in the program list as the title of the program item. Next, enter

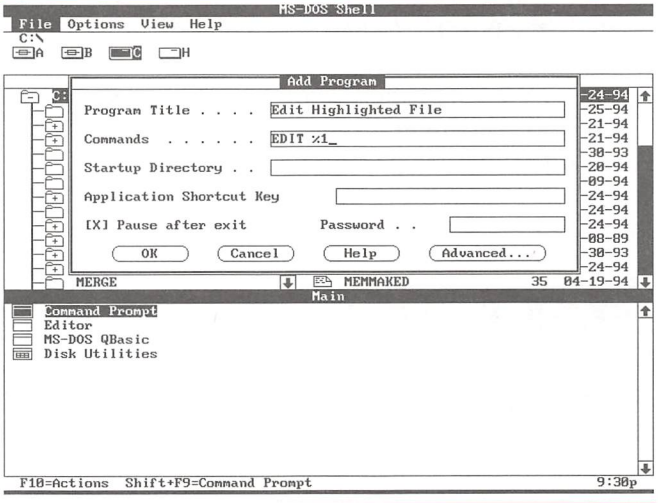
```
EDIT %1
```

in the Commands text box. The %1 parameter creates a dialog box that would normally prompt you for the name of the file you want to edit. Leave the *Pause after exit* option selected. In the next step, you'll add a parameter that selects the highlighted file. At this point, the Add Program dialog box should look like the one shown in Figure C. Now, press [Enter], and you'll see a second Add Program dialog box.

The second Add Program dialog box asks for text to display in the dialog box created with the %1 parameter. Even more vital to our technique, this Add Program dialog box lets you enter a default parameter that will appear in the prompt dialog box whenever you choose the program item. You can use either of two parameters:

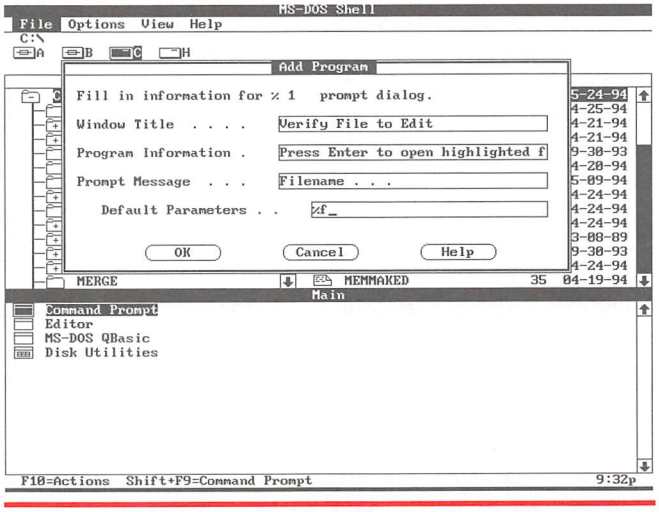
- %f—to specify the file that's highlighted when you start the program
- %l—to specify the parameter that was used the last time you ran the program

Figure C



You specify in the Add Program dialog box the program item's name and its commands.

Figure D



In the second Add Program dialog box you specify information that will appear in the prompt dialog box when you run the program.

To create the Edit Highlighted File program item, type into the Add Program dialog box the information in the dialog box shown in Figure D. The information in the Window Title box displays at the top of the prompt dialog box. The Program Information box can contain up to 106 characters. Ours contains the text

```
Press Enter to open highlighted file in the Editor. Or,
    backspace and type a different filename.
```

The Prompt Message box can contain up to 18 characters of text, which you use to name the text box that holds the name of the highlighted file. We specified %f in the Default Parameters box to place the highlighted file's name in the Filename text box.

Be sure to use the [Tab] or ↓ key to move from one box to the next—if you press [Enter], the Shell will assume you're finished and will close the Add Program dialog box. If this happens, you can reopen it by highlighting the Edit Highlighted File program item, selecting the Properties... command from the File menu, and pressing [Enter] when you see the Program Item Properties dialog box. This box is the same as the Add Program dialog box (its name is different because you opened it with a different command).

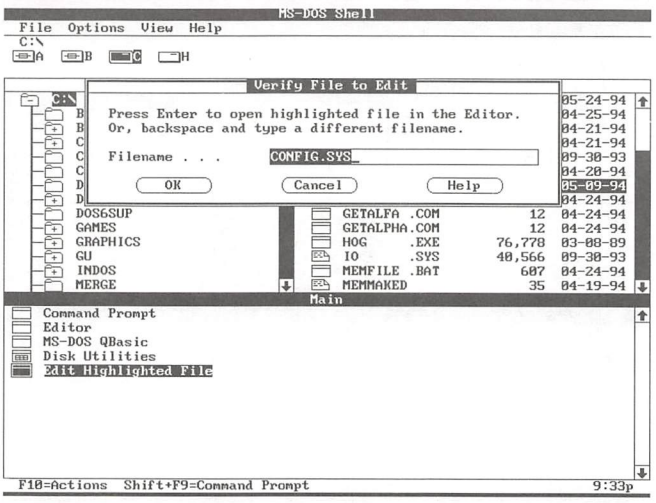
When you've filled in all the information, press [Enter] or click OK to close the dialog box. The Main program list now shows your new icon and entry—Edit Highlighted File. Let's try it out.

An example

We'll use our new program item to open the CONFIG.SYS file from the DOS Shell. If you're using your keyboard, press [Tab] until the cursor is in the directory tree, use the ↓ and ↑ keys to highlight the C:\ entry, tab to the file list, use ↓ to highlight the CONFIG.SYS filename, tab to the program list, use ↓ to highlight the Edit Highlighted File entry, and press [Enter]. If you're using a mouse, click on the C:\ entry in the directory tree, click on the CONFIG.SYS filename in the file list, and then double-click the Edit Highlighted File item in the program list. When you do, the Verify File to Edit dialog box will open.

As you can see in Figure E, the Verify File to Edit box contains all the information you entered into the Add Program dialog box we showed you in Figure D. The highlighted file, CONFIG.SYS, appears in the Filename text box. To open the named file in the Editor, simply press [Enter] or click OK.

Figure E



When you run the Edit Highlighted File program item, the file you highlighted automatically appears in the prompt box.

The DOS Editor now opens with the CONFIG.SYS file in the window. Simply make your changes and press [Alt]F and then X to exit the Editor. Answer <Yes> when the prompt asks whether you want to save your changes. Next you'll see the message

```
Press any key to return to MS-DOS Shell....
```

When you press a key, you return to the Shell, where you can continue with your work.

Conclusion

The DOS Shell provides a number of ways you can run the DOS Editor, but all of them require you to type the name of the file you want to open. In this article, we showed you how to create a program item that lets you highlight a file to open in the Editor. ◆

VAN WOLVERTON

VERSION 5.0 & 6.x

Controlling your DOS environment

Have you checked your environment lately? No, not the temperature or air quality—not *that* environment. I'm talking about your DOS environment—the area of memory where DOS stores information available to any program. This information doesn't change, no matter which program is running, unless you or a program explicitly change it.

The confusion about the term *environment* illuminates one of the more vexing problems in our dealings with computers: the use of familiar words to mean something quite different from their normal connotations. *Environment* is but one example. (Have you actually "booted" your computer lately?)

The environment provides a means of passing information to DOS or application programs with

minimal effort. You're familiar with two types of information stored in the environment: the prompt definition and the command path (we talked about the latter in last month's article "Charting a Path to Commands"). Your system's DOS environment probably includes quite a bit more than that.

Environment variables

In last month's column, we showed you a couple of batch files that use something called an *environment variable* to work with the command path. An environment variable is one item of information that DOS stores in the environment. It consists of a name and a value separated by an equal sign.

DOS includes a single command to manage the environment: the SET command. We can use the SET command to create or change the value of an environment variable, to delete an environment variable, or to display the contents of the environment.

To see what's in the environment, type *set* with no parameters. When you do, DOS displays all the environment variables:

```
PATH=C:\DOS;C:\;C:\BATCH;C:\MENU;C:\WORD;C:\WINDOWS
PROMPT=$p$g
COMSPEC=C:\DOS\COMMAND.COM
TEMP=c:\temp
NET=yes
NU=c:\nu
PCPLUS=c:\pcplus
PLAYER1=Fred
```

Your display is different than mine, of course, but at least a couple of the variables shown here—PATH and PROMPT—should appear in your display. PATH, as we saw last month, defines the command path. PROMPT, not surprisingly, defines the DOS prompt. These two variables are present in virtually every DOS environment, although they probably aren't the first two in your environment, and their values are almost certainly different.

The COMSPEC variable tells DOS where to find the *command interpreter*, the program that checks what you type and decides which command to run. Most DOS users rely on the default command interpreter, COMMAND.COM. The TEMP variable tells Windows where to store temporary files. The NET environment variable comes from a directive in my CONFIG.SYS. I use DOS 6's multiple configuration option in CONFIG.SYS. When I respond to the prompt that asks whether I want to start the network, NET is set to either *yes* or *no*.

The NU and PCPLUS variables contain paths to the directories that hold the Norton Utilities and Procomm Plus programs, respectively. During installation, these programs placed SET commands in AUTOEXEC.BAT. These SET commands create the NU and PCPLUS variables each time the computer boots.

The case of the shrinking environment

The last environment variable illustrates a problem only hinted at by the NU and PCPLUS variables. PLAYER1 is an environment variable used by a game I deleted from this computer a couple of years ago. Its installation program, too, put in AUTOEXEC.BAT a SET command that creates this environment variable. When I deleted the game, I forgot to delete the SET command.

It's fairly common for programs to create one or more environment variables this way. However, you don't always keep programs forever. You typically remove a program from your computer by deleting its program and data files. When you do, you should also delete any SET commands the program created in AUTOEXEC.BAT, because each variable takes up some of the limited space available in the environment. However, you can't just delete SET commands that create environment variables you don't recognize, because it isn't always easy to know which program uses which variable.

PCPLUS, for example, is pretty easy to figure out because Procomm Plus uses PCPLUS as the name for several of its associated files, but we might forget that NU came from Norton Utilities. Even though it's simple to determine that a game uses PLAYER1, it might be hard to decide which game if you've had several games installed on your computer.

You might find several unfamiliar variables that you'd like to delete from your environment. If so, you might consider opening your AUTOEXEC.BAT file in the DOS Editor and erasing the SET commands that create the variables. However, you can use the REM command to effectively remove the SET commands without wiping them out completely. All you do is place the word *REM* at the beginning of each line containing a SET command that creates an environment variable you'd like to eliminate. Use the system for a week or two to make sure that all your programs work properly.

If a program won't run or runs incorrectly, you may have to use trial and error to find the environment variable the program uses. Edit AUTOEXEC.BAT again and remove *REM* from one of the SET commands—choose one at random if none of the variables seem likely candidates. Then restart your system and try the program again. If the problem persists, edit AUTOEXEC.BAT again, retype *REM* at the beginning of the SET command, remove *REM* from the beginning of another SET command, then restart and try the program again. Continue until you find the environment variable causing the problem. When you find

the culprit, leave its SET command in AUTOEXEC.BAT and delete the others.

Creating and deleting an environment variable

Except for the PATH and PROMPT variables, which define the command path and prompt, DOS doesn't pay any attention to environment variables. The other environment variables exist so that programs can use them. As we've seen, some programs also define variables for their use. You may define any variables you like, as long as their names differ from variables other programs use.

When you define an environment variable with the SET command, DOS converts its name to uppercase letters, no matter how you enter it. However, DOS leaves the environment variable's value—that's the text that follows the equal sign—just as you type it. Therefore, uppercase and lowercase letters represent different values. For example, either of these commands creates an environment variable with the same value:

```
set player1=Fred   set PLAYER1=Fred
```

Once created, a variable remains in the environment until it's changed or deleted by a SET command.

You can also use SET to delete an environment variable by following the name with nothing but an equal sign. For example, you could delete the preceding environment variable by typing

```
set player1=
```

You can use the SET command to change the command path or prompt, but it's a bit quicker to use the DOS PATH and PROMPT commands to do so.

Saving your path and prompt

If you experiment with environment variables, you might accidentally delete either your command path or prompt definition—it's a pain to retype them if they're long. We'll show you a couple of ways to create batch files that restore your current command path and prompt.

When you type just *path*, DOS displays *PATH=* followed by the names of the directories in the command path, separated by semicolons. This display is the proper form for a PATH command, so you can create a batch file that restores your current command path by redirecting the output of the PATH command to a file we named PATHRS.BAT:

```
path > pathrs.bat
```

Now all you do to restore the command path is type *PATHRS*.

You have to do a little more work to create a batch file to restore your current prompt definition, because there isn't a command that tells DOS to display the current prompt definition. But the SET command displays all the environment variables, and the PROMPT command contains the prompt definition. So, you can still capture the PROMPT definition by piping the output of the SET command to a FIND command that searches for PROMPT, then redirecting the output of the FIND command to a file named PROMPTRS.BAT:

```
set | find "PROMPT" > promptrs.bat
```

Now you can experiment freely without having to retype either a PATH or PROMPT command.

Changing the environment size

The default size for the DOS environment is 160 bytes. If you define a long command path or an elegant prompt, you can quickly use up all the space available for the environment. If you try to create an environment variable—or specify a new value for an existing variable—that would exceed the available space, DOS creates as much as it can, then replies *Out of environment space*. This circumstance results in a partial assignment, which is never correct, so you have to delete the variable entirely.

You can tell DOS to set aside more memory for the environment by putting in CONFIG.SYS a SHELL command that includes the path name of the command interpreter, the path to the directory that contains the command interpreter (most likely C:\DOS), and the /E (for *environment*) and /P (for *permanent*) parameters. When you use the /E switch, follow it with a colon and the number of bytes you want to set aside for the environment. For example, the following SHELL command in CONFIG.SYS sets aside 512 bytes for the environment:

```
shell=c:\dos\command.com c:\dos\ /e:512 /p
```

An environment size of 512 bytes should be large enough, unless you have an exceptionally long command path and prompt definition or unless many programs have created environment variables for their use. If you find that you still get the *Out of environment space* message, increase the size even more.

Conclusion

You don't have to think about the environment very often, but it's an important feature of DOS that you should periodically check and know how to manage.

In this article, we introduced environment variables, described how programs create them, and showed how you can create, delete, and change them. We also showed you how to change the size of your environment.

Contributing editor Van Wolverton is the author of the best-selling books Running MS-DOS *and* Supercharging MS-DOS. *Van, who has worked for IBM and Intel, lives in Alberton, Montana.* ◆

VERSION 5.0 & 6.x

Opening and closing files in the MS-DOS Editor

The MS-DOS Editor is a simple, basic editor—no bells or whistles—you can use for creating text files such as batch files and your CONFIG.SYS file. However, if you take the time to explore the Editor's options, you'll find that it's actually pretty sophisticated. In this article, we'll introduce you to the basic functions of the MS-DOS Editor: opening and closing files. In future issues, we'll explore some of its more advanced features.

Opening an existing file from the DOS prompt

There are a number of ways to open an existing file in the Editor. Probably the easiest way to open a file is to type the filename and, if appropriate, its path when you enter the command that opens the Editor. For example, you can open the file named BETTY.TXT, stored in the BEDROCK directory, by entering the command

```
C:\>edit bedrock\betty.txt
```

You can also open the BETTY.TXT file by switching to the BEDROCK directory and entering the command

```
C:\BEDROCK>edit betty.txt
```

The Editor will place the file's contents in the edit window and its name at the top of the window.

Creating, naming, and opening a file from the DOS prompt

You can create and open a file in the DOS Editor the same way you open an existing file from the command line. You simply type a filename and, if appropriate, the path to the directory in which you want to store it. For example, to create a file named WILMA.TXT in the BEDROCK directory, you enter the command

```
C:\>edit bedrock\wilma.txt
```

If you specify a directory path when you create a file using the EDIT *filename* command, DOS will automatically place the file in the specified directory when you save the file. You can also create a file in a specified directory by invoking the EDIT command from within that directory. In the example above, you could create the WILMA.TXT file by changing to the BEDROCK directory and entering the command

```
C:\BEDROCK>edit wilma.txt
```

Creating and opening an unnamed file from the DOS prompt

You can create an unnamed file in the Editor by typing

```
C:\>edit
```

at the DOS prompt and pressing [Enter]. When you do, the Editor's Welcome screen greets you. Press [Escape] to remove the screen and begin typing your text in the edit window named Untitled. You can name the file at any time by pressing [Alt]F to open the File menu, shown in Figure A on page 8, and then pressing *S* to Save the file or *A* to Save As.... In fact, the Editor won't let you close your Untitled file without prompting you to save and name it.

Let's create a file by typing some text in the Untitled window and saving it. Enter

```
C:\>edit
```

Moving around in the Editor's dialog boxes

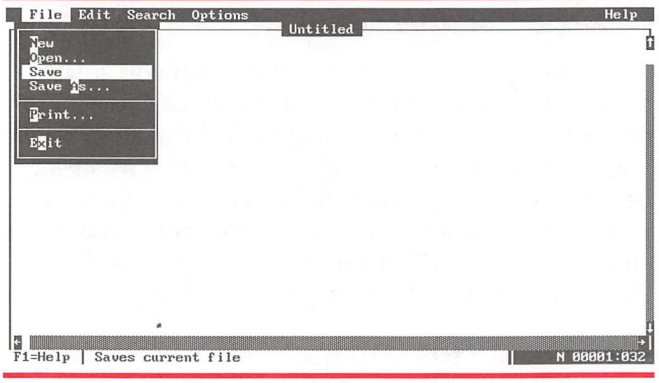
Probably the easiest way to move around in the Editor's dialog boxes is to use a mouse. You simply move the mouse to a window or text box and click once to highlight a selection or click twice to execute your selection. If you want to select an item from a list and the item isn't visible, you can click the scroll bar or scroll arrow at the right or bottom of the list to bring more items into view.

If you use only the keyboard, navigating dialog boxes is a little more involved. You use the [Tab] key to move from one box, window, or button to the next. You can use [Shift][Tab] to move from a box, window, or button to the previous one. If you're in a window that contains a list, you can use the ↑ and ↓ keys to move from item to item, or you can press the first letter of the file or directory name you want to select. If more than one item starts with the letter you pressed, you use the ↑ and ↓ keys to highlight the item you want. You can use the [Home] and [End] keys to move to the top and bottom of a list. Finally, you select an item by pressing [Enter].

You can also use the [Alt] key to navigate dialog boxes using only the keyboard. When a dialog box is open, you press [Alt] and the Editor highlights one letter in the name of each box or window. To move the cursor to a box or window, you simply hold down the [Alt] key and press the highlighted letter. Then type the first letter of your filename and/or use the arrow keys to make your selection.

to open the Editor, press [Escape], and type *Yabba dabba doo* in the window. Now, save and name the file by pressing [Alt]F, pressing S, typing *fred.txt* in the Save dialog box, and pressing [Enter].

Figure A



You save an unnamed file by selecting the Save... or Save As... option on the File menu.

Selecting a file in the Editor's Open dialog box

You can open a file that already exists by entering

```
C:\>edit
```

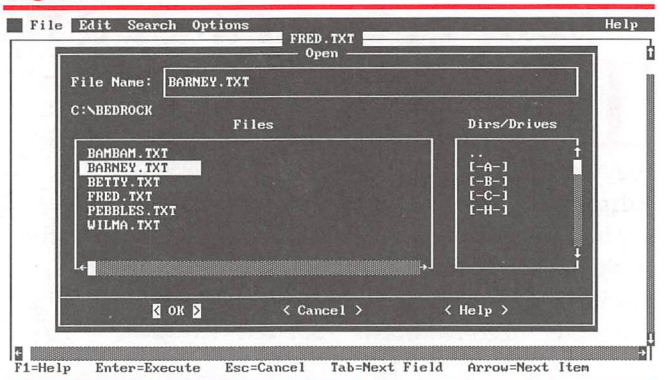
at the command line, pressing [Escape] to bypass the Welcome dialog box, pressing [Alt]F, and selecting the Open... option. Now you can type the path and filename of the file you want to open in the text box. Of course, if you know the filename, you'll save time by entering its name at the DOS prompt.

Specifying a file that appears in the Files window

If you don't know the exact name of the file you want to open, or if you don't know which directory it's in, you'll need to search for the file in the Files and Dirs/Drives windows. The default file extension in the File Name text box is TXT. To see files in the current directory that have another extension, press [Backspace] three times to erase *TXT*, type the extension you want, and press [Enter]. You can replace the TXT extension with the wildcard character * to display all the files in the current directory.

To select a file that's listed in the Files window, you first move the cursor to the Files window, then highlight the file. (You might want to read "Moving Around in the Editor's Dialog Boxes" on page 7 to learn how to navigate dialog boxes.) For example, to open the file BARNEY.TXT, you press [Tab] to move the cursor to the Files window and then press ↓ to highlight the filename. As Figure B shows, when you highlight the filename, its name appears in the File Name text box. To select the highlighted file, press

Figure B



When you highlight a file in the Files window of the Open dialog box, its name appears in the File Name text box.

[Enter] or double-click the filename with your mouse, and the file will open in the Edit window.

Specifying a file in a different directory

If the file you want to open isn't listed in the Files window, you can search for it in another directory or on another drive. If you know all or part of the filename, enter it in the File Name text box. Then move the cursor to the Dirs/Drives window and scroll down the list of directories. As you scroll through the list, the File Name text box displays the currently highlighted directory name or drive letter. Move the highlight to the directory you want and press [Enter] or double-click the directory name.

The Files window now shows all the files that match the specification in the File Name text box. Move the cursor back to the Files window and select the file you want to edit. Press [Enter] or double-click the file's name to open the file.

If the file you want isn't in the directory you opened, you can move back to the parent directory by selecting the .. entry in the Dirs/Drives window. Then you simply highlight and select another directory to search.

An alternative to using the Open dialog box

As you can well imagine after reading the steps we just outlined, searching for your file using the Open dialog box can be as tedious as breaking rocks for a living. If you're at the DOS prompt and you need to find the exact location and name of the file you want to edit, try using the directory command

```
C:\>dir /s filespec | more
```

where *filespec* is as much of the filename as you know. You can use a wildcard to represent any portion of the filename that you don't know. Once you know the exact path and filename, you can open the file in the Editor from the DOS prompt by using the command

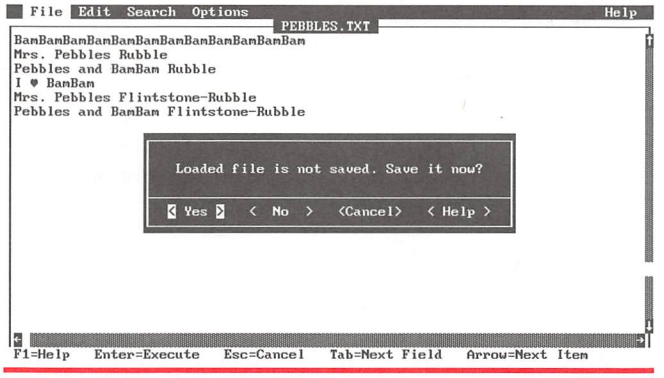
```
edit filepath_&_name
```

Opening a different file once you're in the Editor

Once you're in the Editor, you can create a new file or open an existing one. To create a file, press [Alt]F and then press N to select the New option. To open an existing file, press [Alt]F and then press O to select the Open... option. Then type the name of the directory and file you want to open or use the Files and Dirs/Drives windows to locate and select the file.

The Editor won't let you close an unsaved file without first prompting you to save it. If you haven't saved the file that's currently open in the Edit window, you'll see the prompt shown in Figure C. When you answer Yes (or No, if you don't want to save the changes to the current file), DOS will save the current file, close it, and open the second file.

Figure C



The Editor always prompts you to save your file before closing it.

Saving a file

You should always save your files on a regular basis, just in case your system locks up or you experience a power outage. To save an existing file, use the Save command on the File menu by pressing [Alt]F, S. To save and name an Untitled file, press [Alt]F, S and enter a name for the file in the Save dialog box. Be sure to include a path with the filename if you want to save it in a directory other than the current one.

To save an existing file under a new name, you press [Alt]F, A to choose the Save As... command on the File menu. In the Save As dialog box, type the new filename and the drive and directory in which you want to save the file if it's different from the current one.

Exiting the Editor

We'll assume that, at this point, you've created or opened a file, made a few changes to it, and are ready to leave the Editor. You leave the Editor by pressing [Alt]F and then X to choose the Exit command on the File menu. If you've edited the current file without saving it and you try to leave the Editor, you'll see the prompt we showed you in Figure C.

Conclusion

The MS-DOS Editor is very useful for creating simple text files. There are a variety of ways you can open and close new and existing files in the Editor—as we showed you in this article. ◆

BATCH FILE CHOICES

VERSION 5.0

Soliciting user input in DOS 5

Last month, we talked about adding the [Escape] key as an option for the DOS 6 CHOICE command ("Adding an Escape Hatch for DOS 6's CHOICE Command"). Unfortunately, DOS 5 doesn't offer the CHOICE command or any other built-in means of soliciting user input in a batch file. However, you can add a lot of flexibility to your batch files with a program that pauses your batch file until you press a key and then executes a command based on that keypress.

In this article, we'll present GETALPHA.COM, a DEBUG program you can use in any batch file where you want to let the user make a choice. The GETALPHA program is just one of several public domain programs you can use for this purpose. We chose this particular program for three main reasons:

- GETALPHA isn't case sensitive, so it responds the same whether you type a letter in uppercase or lowercase.
- GETALPHA lets you use the [Escape] key as one of the responses.
- GETALPHA sets the value of the ERRORLEVEL variable to a number corresponding to the letter's place in the alphabet instead of the letter's ASCII code. For example, ERRORLEVEL for the letter A is 1 instead of 65. (We gave a general overview of using the ERRORLEVEL variable in a batch file in last month's article "Using ERRORLEVEL to Make Decisions in a Batch File.")

Once you add GETALPHA.COM to your arsenal, you'll find it brings a whole new dimension to writing batch files. First we'll look at how to create GETALPHA.COM; then we'll show you how to use it in a batch file.

Creating GETALPHA.COM

To create GETALPHA.COM, you create the script file GETALPHA.SCR. After creating the script file, you run the DEBUG utility, using the script file as input. When DEBUG runs the commands in GETALPHA.SCR, it creates GETALPHA.COM and then returns to the DOS prompt.

You can create the script file in the DOS Editor or any word processor that creates a text file without adding special formatting codes. However, the quickest way to create it is to copy lines from the console to the file. To create GETALPHA.SCR in this manner, type the following lines exactly as shown:

```
C:\>copy con getalpha.scr
E 100 B4 00 CD 16 24 DF 24 3F B4 4C CD 21
N GETALPHA.COM

RCX
C
W
Q
<[F6]> <[Enter]>
```

Be sure to include the blank line before the RCX command—the program won't compile correctly if you don't include it. After you press [F6] and then [Enter], DOS will respond with the message *1 file(s) copied*. If you discover an error on a previous line, you'll have to edit the file in the DOS Editor or recreate the file using the COPY CON technique.

You create GETALPHA.COM from the script file by running GETALPHA.SCR through DOS's DEBUG program. The command

```
C:\>debug < getalpha.scr
```

starts DEBUG and directs into it the information from GETALPHA.SCR. Doing so creates the program GETALPHA.COM. That's all there is to it. Once you've created GETALPHA.COM, you can use it in any batch file, as long as the program is in the same directory as your batch file or is in a directory that's listed in your PATH statement. Now that you've created GETALPHA.COM, let's look at how you use it in a batch file.

Using GETALPHA.COM in a batch file

By itself, GETALPHA.COM is fairly useless. You can run the program from the DOS prompt, but nothing happens until you press a key. When you press a key, DOS presents another prompt. You must use the GETALPHA program in a batch file, and your batch file should contain these components:

- A prompt or message that lists the available choices
- The GETALPHA command, which pauses the batch file until you press a key and then assigns the code for that key to the ERRORLEVEL variable
- A block of IF...GOTO statements that test the contents of the ERRORLEVEL variable and send the batch file to the appropriate branch
- A series of action blocks—labeled branches that contain commands appropriate to the key you pressed

Now, let's create a sample batch file that combines all these elements. We'll look at each element in more detail as we discuss the batch file.

A sample batch file

Since we're creating the batch file for demonstration purposes, let's have some fun with it. If you'd like to create our BASEBALL.BAT batch file, simply open the DOS Editor by entering the command

```
C:\>edit baseball.bat
```

on the command line. Then, type the batch file commands shown in Figure A. Finally, save the file and close the Editor by pressing [Alt]F, X and answering <Yes> to the message *Loaded file is not saved. Save it now?*

Now let's look at how BASEBALL.BAT works. Following the initial @ECHO OFF and REM statements, the batch file uses a series of ECHO statements to display a menu that lists all the valid choices.

Next, the batch file calls GETALPHA.COM, which pauses the batch file until you press a key. Because GETALPHA.COM is an executable program, you can enter its name into your batch file without a CALL or GOTO command. At this point, the GETALPHA program controls the batch file. Once you press a key, GETALPHA assigns that key's code to the ERRORLEVEL variable—the code for *A* or *a* is 1, *B* or *b* is 2, and so on through *Z* or *z*, which is 26, and [Escape], which is 27.

Now the batch file resumes control and checks the code stored in ERRORLEVEL against the numbers in the block of IF statements that follows. We use the syntax

```
if errorlevel x if not errorlevel x+1 goto LABEL
```

so the batch file will branch to *LABEL* if the ERRORLEVEL code is exactly *x*. (In this example, *x* is the code number and *LABEL* is the name of the branch containing the appropriate commands.) If we don't include the IF NOT statement, the batch file will branch to *LABEL* if the ERRORLEVEL code is equal to or greater than *x*.

Since we're checking for nonconsecutive keys (C, G, R, and [Escape]), we need to specify the exact ERRORLEVEL code. If we didn't do this, the batch file would go to the RUTH branch if we accidentally pressed the *H* key.

Figure A

```
@echo off
rem BASEBALL.BAT demonstrates how to use GETALPHA.COM
rem in a batch file.
echo.
echo Press the letter next to the player's name to see facts
echo about the Baseball Hall of Famer:
echo.
echo C   Ty Cobb
echo G   Lou Gehrig
echo R   Babe Ruth
echo.
echo Or press [Escape] to return to the prompt.

:PROG
getalpha

if errorlevel 27 if not errorlevel 28 goto END
if errorlevel 18 if not errorlevel 19 goto RUTH
if errorlevel 7 if not errorlevel 8 goto GEHRIG
if errorlevel 3 if not errorlevel 4 goto COBB
if errorlevel 1 goto ERROR

:COBB
echo.
echo Tyrus R. Cobb (1886-1961) played at Detroit and
echo Philadelphia 1905-1928. Retired with 4191 hits.
goto END

:GEHRIG
echo.
echo Henry L. Gehrig (1903-1941) played with the New York
echo Yankees 1923-1939. Lifetime batting average of .340.
goto END

:RUTH
echo.
echo George Herman Ruth (1895-1948) played at Boston and New
echo York 1915-1935. Called the "Greatest drawing card in
echo the history of baseball."
goto END

:ERROR
echo You pressed an illegal key.
echo Select a letter from the above list.
goto PROG

:END
```

BASEBALL.BAT uses the GETALPHA program to solicit input from the user.

Because the first four IF statements check for exact ERRORLEVEL codes, we can arrange them in any order. However, IF ERRORLEVEL 1... must be last so that the batch file can check for invalid keypresses. For example, if you type *E* at the menu, the batch file compares the ERRORLEVEL code for E—which is 5—to the first IF statement. The batch file determines that the current ERRORLEVEL value doesn't equal 27, so it checks the next IF statement. After the batch file determines that the code stored in ERRORLEVEL doesn't match any of the first four IF statements, it evaluates the last statement:

```
if errorlevel 1 goto ERROR
```

Since ERRORLEVEL contains 5 and since 5 is greater than 1, the batch file recognizes that this statement is true and jumps to the :ERROR branch. If we'd placed the last IF statements first, the batch file would always jump to the ERROR branch, because every possible ERRORLEVEL code—including the valid ones—is equal to or greater than 1.

Once the batch file finds an IF statement that tests True, it jumps to the action block specified by the true IF statement's GOTO command. Each labeled branch contains commands to execute—in this case, ECHO commands. Most of the action blocks end with the GOTO END command, which forces the batch file to skip over any action blocks below it and jump to the :END label. Thus, the batch file executes only the commands in the labeled branch.

However, the :ERROR branch lets you know you pressed an invalid key and prompts you to select a key from the list. After displaying the prompt, the :ERROR branch sends the batch file back to the :PROG branch. The :PROG branch executes GETALPHA again, which pauses the batch file until you press a valid key.

The final command in the batch file is the :END label. There are no commands associated with this branch, so when the batch file reaches the :END label, it quits and returns control to the DOS prompt.

The batch file contains all four of the components we set up in the section "Using GETALPHA.COM in a Batch File." You should use this batch file only to familiarize yourself with setting up a batch file that uses the GETALPHA program—unless, of course, you decide to turn it into a baseball trivia game.

Notes

GETALPHA.COM is one of several available public domain and shareware programs you can use to solicit user input. We already mentioned its advantages. It has one main disadvantage: GETALPHA.COM recognizes only letters and the [Escape] key. You can't use the number keys or other character keys as options when you use the GETALPHA program in a batch file. If you press one of the keys GETALPHA.COM doesn't recognize, you'll get unexpected results.

If you want a program that accepts numbers and other keyboard characters, you can use the REPLY.COM program we presented in the August 1993 article "The Most Efficient Way to Solicit User Input in DOS 5 Batch Files." However, if you use REPLY.COM, you must test the ASCII code of the valid keys. Also, REPLY.COM is case-sensitive, so you must test for both uppercase and lowercase letters. For example, if A is one of the valid keys, you need two IF statements to test for both *A* (ASCII code 65) and *a* (ASCII code 97):

```
if errorlevel 97 if not errorlevel 98 goto LABEL
if errorlevel 65 if not errorlevel 66 goto LABEL
```

Summing up

DOS 5 offers no means of soliciting input in a batch file. In this article, we presented GETALPHA.COM, a DEBUG utility that fulfills this need in DOS 5, and showed you a batch file to use the GETALPHA program. ◆

SECOND CLASS MAIL

Microsoft Technical Support
(206) 454-2030

LETTERS

Why does SET PATH in CONFIG.SYS truncate the path after 132 characters?

I teach classes on several software programs, and I've done everything I can think of to shorten my PATH statement. I was very pleased to find your April article "Extending Your Path Beyond the 122-Character Limit." However, your technique doesn't seem to work the way you say it will. When I try to display my 184-character PATH statement, my system cuts it off after 132 characters. I've tried using both the SET command and the PATH command at the DOS prompt, but all I get are the first 132 characters. Can you enlighten me?

Margaret McNulty
Denver, Colorado

The April article showed how to extend your PATH statement beyond 127 characters in DOS 6.x. You simply place the PATH statement in your CONFIG.SYS file using the SET command. By placing the PATH statement in your CONFIG.SYS file, you bypass the 127-character limit DOS places on commands entered from the command line or from a batch file—AUTOEXEC.BAT, in this case. However, the DOS Editor places an upper limit on the length of your PATH statement—the maximum length of a single line in the DOS Editor is 256 characters.

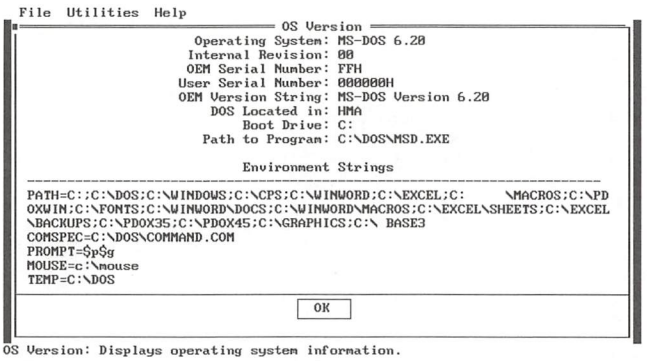
In trying to display the PATH statement, Ms. McNulty has run up against that 127-character limit from a slightly different angle. When you issue the SET and PATH commands from the DOS prompt, you essentially ask COMMAND.COM to display the contents of the PATH variable. Coincidentally, COMMAND.COM is unable to display more than the first 127 characters of PATH in addition to the 5 characters in the string *PATH=*. These 132 characters constitute the length of the PATH statement segment you can display with the SET and PATH commands.

Our technique *does* work. Test your new PATH statement by executing a command stored only in a directory near the end of your PATH statement. For example, if you placed your 200-character PATH statement in CONFIG.SYS, and if C:\DBASE3 is the last directory listed, you can verify that your PATH statement works by saving CONFIG.SYS, pressing [Ctrl][Alt][Delete] to reboot your computer, and entering

```
C:>dbase
```

at the prompt. If your PATH statement works, you should see the dBASE III PLUS opening screen shortly.

Figure A



View your complete PATH statement using the OS Version... option of the MSD utility.

Incidentally, you can view your complete PATH statement in the DOS 6.x MSD utility. Open Microsoft Diagnostics by entering

```
C:>msd
```

at the DOS prompt. Then, at the MSD main screen, press O to select the OS Version... option. When you do, MSD displays the screen shown in Figure A. All your current environment settings display in the bottom half of the screen. ◆

Printed in the USA
This journal is printed on recycled and recyclable paper.
50% recycled fiber including 10% post-consumer waste paper